Trabajo Practico 2 – Sockets

Redes y Comunicaciones

Integrantes del Grupo A

- Lourdes Noelia Crespo Barreira
- Iñaki Martínez Ludueña
- Santiago Cano
- Santiago Sánchez
- Ariel Contreras

Repositorio de GitHub - TP Sockets

Índice

Consigna

Una empresa necesita que se desarrolle un sistema de sockets compuesto por 1 (un) servidor y 2 (dos) clientes, donde uno de los clientes debe ser en el mismo lenguaje de programación que el servidor y el otro en un lenguaje de programación distinto, ambos generando una conexión con un cliente.

Los requerimientos son:

1. Desarrollar un servidor en uno de los siguientes lenguajes: C, Java o Python, utilizando sockets e implementando las siguientes funcionalidades:

   - Un generador de nombres de usuario que cree el nombre de usuario basado en la longitud especificada por el cliente, pero debe validar que no sea menor a 5 ni mayor a 20. Además, este debe incluir al menos una vocal y una consonante. No puede incluir números.

   - Un generador de direcciones de correo electrónico. El servidor deberá generar una dirección de correo electrónico aleatoria basada en el nombre de usuario proporcionado por el cliente. Además, la dirección de correo electrónico se debe validar, es decir, debe haber un verificador que verifique si es una posible dirección de correo electrónico.

   Ejemplo: Nombre de usuario: EduardoGomez Correo electrónico: eduardogomez@gmail.com (ESTE ES UN EJEMPLO VÁLIDO)

   Nombre de usuario: EduardoGomez Correo electrónico: eduargogomez@mail.com (ESTE ES UN EJEMPLO INVÁLIDO ya que mail.com NO es un dominio válido. El verificador debe validar eso, que sea un correo válido)

   En este caso solo vamos a usar 2 dominios válidos: @gmail.com y @hotmail.com

2. Desarrollar un primer cliente en un lenguaje de preferencia el cual se conectará con el servidor y presentará un menú que permitirá al usuario elegir entre generar un nombre de usuario o una dirección de correo electrónico. Para la opción de dirección de correo electrónico, el cliente debe permitir al usuario ingresar su nombre de usuario y crear el correo electrónico al azar en base a ese nombre de usuario (Recordar que debe ser válido el dominio). El cliente debe mostrar la respuesta generada por el servidor, incluyendo mensajes de error en caso de validación.
3. Desarrollar un segundo cliente en un lenguaje diferente, la funcionalidad debe ser la misma que el primer cliente, pero se implementará en un lenguaje diferente

Ejemplos:

- Si el primer cliente utiliza C, el segundo cliente puede utilizar Python.
- **Si el primer cliente utiliza Python, el segundo cliente puede utilizar Java. (Usamos este)**
- Si el primer cliente utiliza Java, el segundo cliente puede utilizar Go.

Notas importantes

- El servidor debe estar en ejecución antes de iniciar cualquiera de los clientes
- Los clientes se conectan automáticamente al servidor en localhost (127.0.0.1)
- La comunicación se realiza a través del puerto 65432

Entorno de ejecución

- **Servidor:** Python usando la librería socket
- **Cliente 1:** Python, archivo client.py
- **Cliente 2:** Java, archivo JavaClient.java
- **IDE:** VSCode

Instrucciones de ejecución

En la carpeta del trabajo practico, donde están los archivos, abrir una terminal y ejecutar.

1. Iniciar el servidor

```
PS D:\UNLa\2025\1er Cuatrimestre\Redes y Comunicaciones\TP 1> python server.py
[INICIANDO] Servidor iniciado...
[INICIANDO] Servidor iniciado en 127.0.0.1:65432
```

2. Ejecutar el cliente de Python - En una terminal nueva, ejecutar.

```
PS D:\UNLa\2025\1er Cuatrimestre\Redes y Comunicaciones\TP 1> python .\client.py
```

3. Ejecutar el cliente de Java - En una terminal nueva, ejecutar.

```
PS D:\UNLa\2025\1er Cuatrimestre\Redes y Comunicaciones\TP 1> java .\JavaClient.java
```

Pruebas de conexión

Servidor - Cliente 1

```
PS D:\UNLa\2025\1er Cuatrimestre\Redes y Comunicaciones\TP 1> python .\client.py
Conectando al servidor...
Conectado al servidor con éxito.

==== CLIENTE PYTHON ====
1. Generar nombre de usuario
2. Generar email
3. Desconectar
Seleccionar una opción:
```

Servidor - Cliente 2

```
PS D:\UNLa\2025\1er Cuatrimestre\Redes y Comunicaciones\TP 1> java .\JavaClient.java
Conectando al servidor...
Conexión establecida con éxito.

==== CLIENTE JAVA ====
1. Generar nombre de usuario
2. Generar email
3. Desconectar
Seleccione una opción:
```

Servidor

```
PS D:\UNLa\2025\1er Cuatrimestre\Redes y Comunicaciones\TP 1> python .\server.py
[INICIANDO] Servidor iniciado...
[INICIANDO] Servidor iniciado en 127.0.0.1:65432
[NUEVA CONEXIÓN] ('127.0.0.1', 65376) conectado.
[CONEXIONES ACTIVAS] 1
[NUEVA CONEXIÓN] ('127.0.0.1', 65377) conectado.
[CONEXIONES ACTIVAS] 2
```

Pruebas de generador de nombre de usuario

Servidor - Cliente 1

```
==== CLIENTE PYTHON ====
1. Generar nombre de usuario
2. Generar email
3. Desconectar
Seleccionar una opción: 1
Ingrese su nombre completo (nombre y apellido): Lionel Messi
Enviando comando: USUARIO_GENERAR|Lionel Messi

Respuesta del servidor: Nombre de usuario generado: lionelmessi
```

Servidor - Cliente 2

```
==== CLIENTE JAVA ====
1. Generar nombre de usuario
2. Generar email
3. Desconectar
Seleccione una opción: 1
Ingrese su nombre completo (nombre y apellido): Lionel Andres Messi
Enviando comando: USUARIO_GENERAR|Lionel Andres Messi
Respuesta recibida: Nombre de usuario generado: lionelandres

Respuesta del servidor: Nombre de usuario generado: lionelandres
```

Servidor

```
[CONEXIONES ACTIVAS] 2
[RECIBIDO] ('127.0.0.1', 65376): USUARIO_GENERAR|Lionel Messi
[ENVIADO] ('127.0.0.1', 65376): Nombre de usuario generado: lionelmessi

[RECIBIDO] ('127.0.0.1', 65377): USUARIO_GENERAR|Lionel Andres Messi
[ENVIADO] ('127.0.0.1', 65377): Nombre de usuario generado: lionelandres
```

Pruebas de generador de correo electrónico

Servidor - Cliente 1

```
==== CLIENTE PYTHON ====
1. Generar nombre de usuario
2. Generar email
3. Desconectar
Seleccionar una opción: 2
Generando correo electrónico basado en el nombre de usuario...
Enviando comando: EMAIL

Respuesta del servidor: Email generado: lionelmessi@hotmail.com
```

Servidor - Cliente 2

```
==== CLIENTE JAVA ====
1. Generar nombre de usuario
2. Generar email
3. Desconectar
Seleccione una opción: 2
Generando correo electrónico basado en el nombre de usuario...
Enviando comando: EMAIL
Respuesta recibida: Email generado: lionelandres@gmail.com

Respuesta del servidor: Email generado: lionelandres@gmail.com
```

Servidor

```
[RECIBIDO] ('127.0.0.1', 65376): EMAIL
[ENVIADO] ('127.0.0.1', 65376): Email generado: lionelmessi@hotmail.com

[RECIBIDO] ('127.0.0.1', 65377): EMAIL
[ENVIADO] ('127.0.0.1', 65377): Email generado: lionelandres@gmail.com
```

Casos de error (nombre o email invalido)

Nombre vacío

```
==== CLIENTE PYTHON ====
1. Generar nombre de usuario
2. Generar email
3. Desconectar
Seleccionar una opción: 1
Ingrese su nombre completo (nombre y apellido):
Enviando comando: USUARIO_GENERAR|

Respuesta del servidor: ERROR: Debe ingresar al menos nombre y apellido
```

Nombre con números

```
==== CLIENTE JAVA ====
1. Generar nombre de usuario
2. Generar email
3. Desconectar
Seleccione una opción: 1
Ingrese su nombre completo (nombre y apellido): lionel 10
Enviando comando: USUARIO_GENERAR|lionel 10
Respuesta recibida: ERROR: El nombre no puede contener numeros

Respuesta del servidor: ERROR: El nombre no puede contener numeros
```

Generar email sin antes generar nombre de usuario

```
==== CLIENTE JAVA ====
1. Generar nombre de usuario
2. Generar email
3. Desconectar
Seleccione una opción: 2
Generando correo electrónico basado en el nombre de usuario...
Enviando comando: EMAIL
Respuesta recibida: ERROR: Primero debe generar o validar un nombre de usuario.

Respuesta del servidor: ERROR: Primero debe generar o validar un nombre de usuario.
```